في هذا الدرس كيفية التعامل مع قواعد البيانات بشكل برمجي بدون مساعدة الwizard

حيث ستناول الامور التالية ب:

- إجراء عملية الاتصال مع ملف القاعدة
- إضافة سجلات إلى القاعدة
- حذف سجلات من القاعدة
- تعديل على سجلات موجودة
- إجراء عملية بحث عن سجلات ذات خصائص معينة في القاعدة

ولفعل ذلك انشيء ملف قاعدة بيانات باستخدام برنامج MS Access يحتوي على الجدول التالي:

اسم الحقل	نوع الحقل
TableID	ترقيم تلقائي
ENAME	نص
City	نص
Mobile	نص

نملا الجدول بالمعلومات التالية

EName	City	Mobile
Ammar	Baghdad	07801531323
Auday	Basrah	07801534923
Hadi	Muthanna	07801414565
Husien	Hila	07849734923

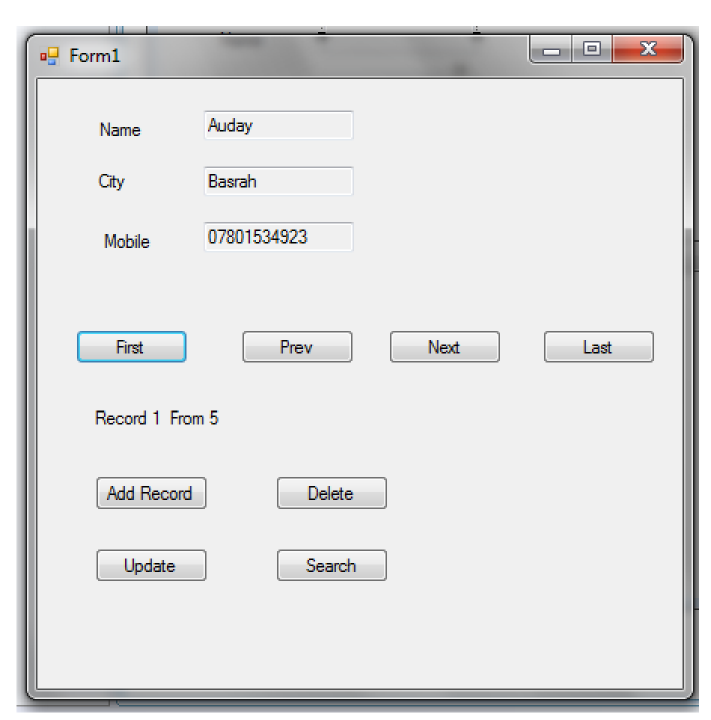


Figure1

ثم نبدأ مشروعاً جديد ومن ثم نقوم بتصميم النموذج بحيث

يبدو كما هو في الشكل (Figure1) ونجعل الخاصية ReadOnly

من textbox1,textbox2,textbox3 تأخذ القيمة true يلي ذلك إضافة الشفرة التالية إلى قسم التصريحات العامة:

```
using System.Data;
using System.Data.OleDb;
```

نقوم بعد ذلك بإضافة الأسطر التالية إلى الصنف الخاصة بالنموذج:

```
public partial class Form1 : Form
    {
        //==================================================
        static string ConStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
            "Data Source=MyData.mdb ";
        OleDbConnection Conn = new OleDbConnection(ConStr);
        DataSet DataSet1 = new DataSet();
        string SQLstr = "Select * from Table1";
        OleDbDataAdapter DataAdapter1;
  //==================================================

                   ….
```

يمثل الحقل ConStr سطر الاتصال أو (Connection String) يحتوي هذا السطر على معلومات تخص مزود قواعد البيانات (Data Base Provider) لاحظ (Provider=Microsoft.Jet.OLEDB.4.0) و كذلك المسار إلى ملف قاعدة البيانات, ففي المثال المذكور أعلاه يقع ملف قاعدة البيانات و الملف التنفيذي للبرنامج في مجلد واحد لذلك لا حاجة ذكر المسار ونكتفي بوضع اسم الملف فقط (Data Source=MyData.mdb) ولكي يكون ملف قاعدة البيانات والملف التنفيذي للبرنامج في مجلد واحد ؛ نضع ملف قاعدة البيانات داخل مجلد البرنامج ومن داخل مجلد البرنامج نجد المجلدBin ثم المجلد Debug ونضع ملف قاعدة البيانات هناك. انظر هذا المسار التقريبي (الأمر يعتمد على جهاز القارئ الكريم)

C:\Users\adel\Documents\Visual Studio 2008\Projects\WindowsFormsApplication13\WindowsFormsApplication13\bin\Debug\MyData.mdb

وقد تستغني عن ذلك بوضع ملف القاعدة في أي مجلد مثلا (D:\1) ونكتفي بالإشارة إليه في سطر الاتصال وعندها يأخذ سطر الاتصال الصورة التالية:

```
static string ConStr=
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=D:\\1\\MyData.mdb ";
```

و إذا أردنا استخدام الطريقتين معا (أي ذكر المسار كاملا مع أن ملف قاعدة البيانات يكون في نفس مجلد الملف التنفيذي) نكتب الكود التالي:

```
static string ConStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
            "Data Source="+Application.StartupPath+"\\MyData.mdb ";
```

بعد ذلك قمنا بتعريف Conn وهو كائن من الصنف OleDbConnection المخصص لإجراء اتصال مع قاعدة البيانات وقمنا بتحديد خصائص عملية الاتصال التي سيقوم بها هذا الكائن بواسطة سطر الاتصال الذي أرسلناه كبارامتر إلى الـconstructure.

و هو أمر فعلناه في السطر

```
        OleDbConnection Conn = new OleDbConnection(ConStr);
```

تقوم الشفرة التالية بتعريف شبكة المعطيات

```
        DataSet DataSet1 = new DataSet();
```

تعتبر شبكة المعطيات الصنف المسؤول عن خزن الجداول ويعمل هذا الكائن في الوضع المنفصل (disconnected mode)

أي أن البيانات الموجودة داخل تعتبر نسخة Snapshot (وليس الأصل)من ما هو موجود داخل القاعدة, يستخدم هذا الأسلوب لتوفير الموارد إذ لولا ذلك للجأنا إلى القاعدة عند كل عملية انتقال من سجل إلى آخر و هو أمر مكلف جدا.

يقوم الحقل SQLstr (وهو عبارة عن سلسلة محارف عادية) بخزن أمر مكتوب بلغة الـSQL

يلي ذلك تعريف كائن من الصنف OleDbDataAdapter و هو الوسيط بين شبكات المعطيات و القاعدة حيث يقوم بجلب المعطيات من القاعدة على شكل جدول ووضعها في شبكة المعطيات .

قم بالنقر مرتين على النموذج (الفورم) واكتب الشفرة التالية:

```
private void Form1_Load(object sender, EventArgs e)
        {
            //******************************
            Conn.Open();
            DataAdapter1 = new OleDbDataAdapter(SQLstr, Conn);
            DataAdapter1.Fill(DataSet1, "Table1");
            Conn.Close();
            //******************************
            textBox1.DataBindings.Add("Text", DataSet1, "Table1.EName");
            textBox2.DataBindings.Add("Text", DataSet1, "Table1.City");
            textBox3.DataBindings.Add("Text", DataSet1, "Table1.Mobile");
            label5.DataBindings.Add("Text", DataSet1, "Table1.TableID");
            //******************************
            ChangePosition();
        }
```

يعمل السطر الأول من الشفرة على فتح قناة اتصال بقاعدة و هي مهمة يضطلع بها المنهج Open (كيف يعرف المنهج المعلومات اللازمة للاتصال؟) بينما يقوم السطر الثاني بتهيئة الوسيط الذي يقوم بدوره بسحب المعلومات اللازمة من القاعدة والتي حددها البارامتر SQLstr , بعد أن تم تامين اتصال له بواسطة البارامتر Conn المفتوح . يعقب هذا الأمر إضافة المعطيات التي جاء بها الكائن DataAdapter1 إلى شبكة المعطيات على شكل جدول (لماذا جلب الكائن المعطيات على شكل جدول ؟) ثم حددنا اسم هذا الجدول بالبارامتر الثاني Table1.

تقوم المجموعة الثانية من الشفرة بربط حقل معين من جدول معين موجود داخل شبكة المعطيات DataSet1 (Table1.EName مثلا) بخاصية (Text في حالتنا هذه) من خواص المكون الذي استدعى المنهج DataBindings.

ملحوظة: اجعل المكون label5 مختفيا وذلك بجعل الخاصية ForceColor تأخذ القيمة Control و اجعل موقعة تحت المكون textBox3 تماما ولا يخفى على القارئ الكريم أن textBox3 سيعرض الحقل Mobile من الجدول (لماذا؟) وافعل نفس الشيء مع الخاصية BackColor من المكون label5 .

يستدعي السطر الأخير المنهج ChangePosition() وهو منهج قمنا بكتابة شفرته لجعل المستخدم يرى رقم السجل الحالي من الجدول Table1 الموجود داخل شبكة المعطيات

```
private void ChangePosition()
        {
            int i1=this.BindingContext[DataSet1, "Table1"].Position+1;
            int i2 = this.BindingContext[DataSet1, "Table1"].Count;
            label1.Text = "Record "+i1.ToString()+"  From "+i2.ToString();
        }
```

يقوم السطر الأول بجلب موقع السجل الحالي من الجدول المحدد (Table1) . طبعا لابد ان تحتوي شبة المعطيات DataSet1 على جدول محدد بالبارامتر الثاني (متى وضعنا هذا الجدول داخل شبكة المعطيات؟) بينما يقوم السطر الثاني بجلب عدد السجلات داخل الجدول المحدد . (ما هو عمل السطر الثالث؟) و هذه هي نتيجة عمل الشفرة السابقة .اضغط على F5 وانظر إلى النتائج.

Record 1 From 5

Figure2

لاحظ ان البرنامج يقوم بعرض السجل الأول فقط حيث أننا لم نتخذ بعد أي إجراءات للتنقل عبر السجلات.

قم بالنقر مرتين على الزر FirstBtn والذي يحوي الخاصية Text ذات القيمة First واكتب الشفرة التالية:

```
private void FirstBtn_Click(object sender, EventArgs e)
        {
            this.BindingContext[DataSet1, "Table1"].Position = 0;
            ChangePosition();
        }
```

ثم قم بالنقر مرتين على الزر NextBtn والذي يحوي الخاصية Text ذات القيمة Next واكتب الشفرة التالية:

```
private void NextBtn_Click(object sender, EventArgs e)
        {
            this.BindingContext[DataSet1, "Table1"].Position += 1;
            ChangePosition();
        }
```

لاحظ أن الخاصية Position أخذت القيمة 0 في الحدث Click الخاص بالزر FirstBtn بينما زادت بمقدار واحد في الحدث Click الخاص بالزر NextBtn.

قم بالنقر على مرتين على الزر PrevBtn والذي يحوي الخاصية Text ذات القيمة Prev واكتب الشفرة التالية:

```
private void PrevBtn_Click(object sender, EventArgs e)
        {
            this.BindingContext[DataSet1, "Table1"].Position -=1 ;
            ChangePosition();
        }
```

قم بالنقر على مرتين على الزر LastBtn والذي يحوي الخاصية Text ذات القيمة Last واكتب الشفرة التالية:

```
private void LastBtn_Click(object sender, EventArgs e)
        {
            this.BindingContext[DataSet1, "Table1"].Position =
                this.BindingContext[DataSet1, "Table1"].Count - 1;
            ChangePosition();
        }
```

لاحظ أننا استدعينا المنهج ChangePosition عند كل مرة قمنا فيها بتغيير موقع السجل الحالي وذلك حتى يتم تعديل رقم السجل الموجود في الخاصية text من المكون label1.

قم بتشغيل البرنامج واستمتع بالتنقل بين السجلات.

في هذا الجزء سنقوم بتنفيذ الإجراءات الخاصة بإضافة سجل جديد لقاعدة البيانات.

ولكي نفعل ذلك نقوم بالاتي:

1. من القائمة الرئيسية نختار البند Project ثم Add Windows Form (Figure3)
2. من النافذة Add New Item اذهب الى Templates واختر البند Windows Forms ثم انقر على الزر ADD (Figure 4)

قم بتنسيق النموذج الجديد بحيث يبدو كما في الصورة (Figure 5)

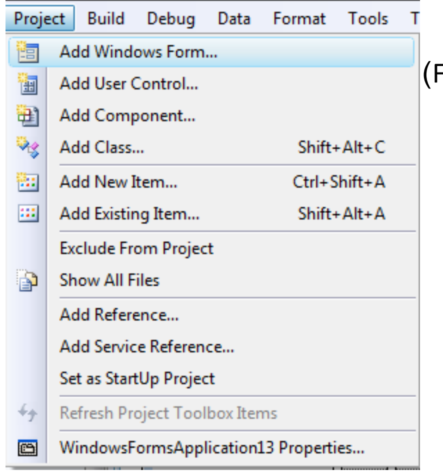


Figure3

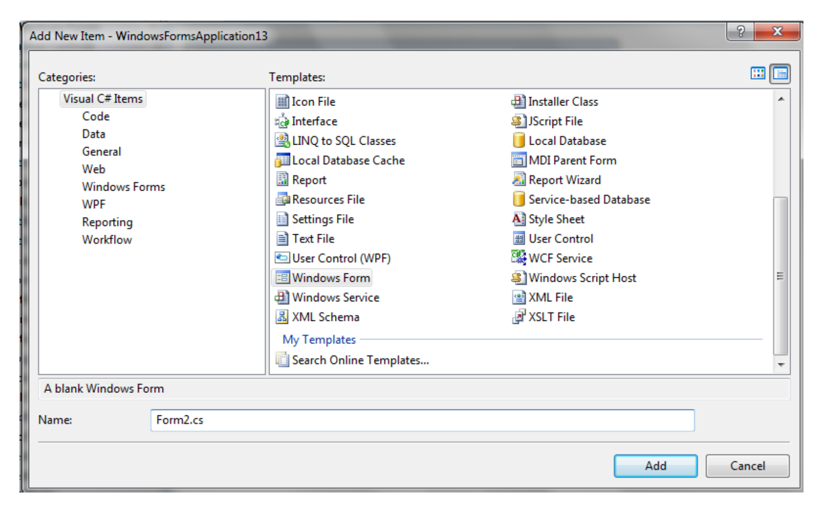


Figure 4

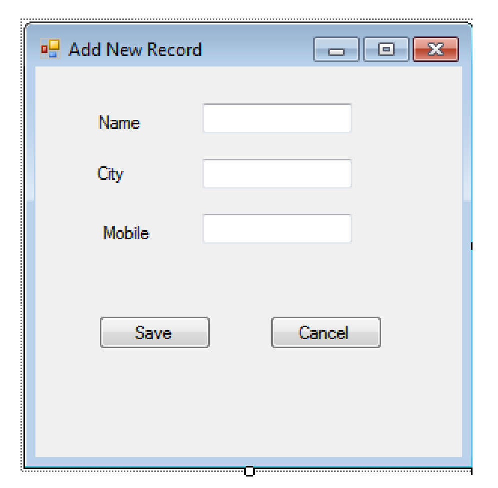


Figure5

والان عد الى النموذج الاول (الفورم الاول) وانقر على الزر Add واكتب الشفرة التالية:

```
private void Add_Click(object sender, EventArgs e)
        {
            Form2 frm2 = new Form2();
            frm2.Show();
            this.Hide();
        }
```

قمنا بإنشاء كائن من نوع Form2 ومن ثم إظهاره .والان قم بإضافة الأسطر التالية الى قسم التصريحات العامة في الشفرة الخاصة بالنموذج Form2

```
using System.Data;
using System.Data.OleDb;
```

لا يخفى على القاريء الكريم انه لإظهار الشفرة الخاصة بأي نموذج ننقر بالزر الأيمن من الفارة على هذا النموذج , ثم من القائمة المنسدلة نختار البند View Code كما في الصورة (Figure 6)

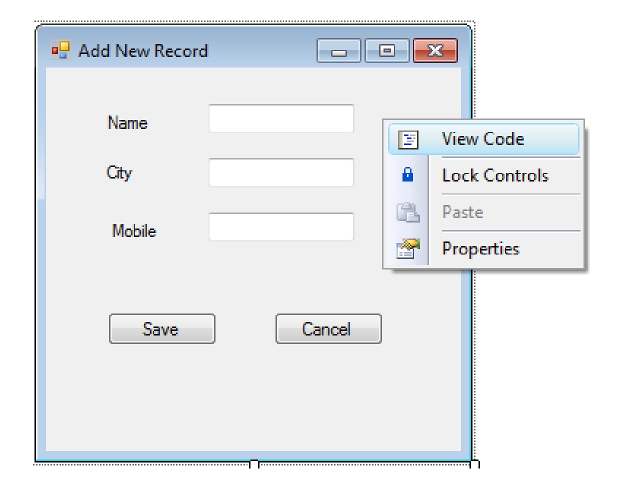


Figure 6

عد إلى النموذج الثاني وقم بإضافة الأسطر التالية إلى الشفرة الخاصة بالنموذج:

```
public partial class Form2 : Form
    {
        static string ConStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
          "Data Source=MyData.mdb ";
        OleDbConnection Conn = new OleDbConnection(ConStr);
       // string SQLstr = "Select * from Table1";
    .....
```

انقر على الزر CancelBtn وهو والذي يحوي الخاصية Text ذات القيمة Cancel واكتب الكود التالي:

```
private void CancelBtn_Click(object sender, EventArgs e)
        {
            Application.OpenForms[0].Show();
            this.Close();
        }
```

يقوم السطر الأول بإظهار النموذج الرئيس من البرنامج حيث يحتفظ الصنف Application بمصفوفة تحتوي على النماذج التي تم إنشاؤها ضمن البرنامج حسب ترتيبها في الانشاء . يقوم السطر الثاني بإغلاق النموذج الثاني

قم بالنقر مرتين على الزر SaveBtn وهو الزر الذي يحتوي على الخاصية Text ذات القيمة Save واكتب الشفرة التالية:

```
private void SaveBtn_Click(object sender, EventArgs e)
        {
            string s = "'";
            OleDbCommand SaveInto = new OleDbCommand();
            SaveInto.Connection = Conn;
            SaveInto.CommandType = CommandType.Text;
            SaveInto.CommandText = "Insert Into Table1(EName,City,Mobile) " +
                  " Values (" + s + textBox1.Text + s + "," + s +
                  textBox2.Text + s + "," + s +
                  textBox3.Text + s + ")";
            //******************
            try
            {
                Conn.Open();
                SaveInto.ExecuteNonQuery();
                Conn.Close();
                MessageBox.Show("Successful Operation");
                Application.OpenForms[0].Show();
                (Application.OpenForms[0] as Form1).RefreshDataSet();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Badly Entered information");
            }
        }
```

ملحوظة : اود أن انوه الى انه للعودة من محرر الشفرة الى مصمم النموذج(Form Designer) ننقر بالزر الايمن للفارة على محرر الشفرة ونختار من القائمة المنسدلة البند View Designer كما هو موضح بالشكل (Figure7)



Figure7

اود فقط أن اذكر القاريء الكريم الصيغة العامة لجملة Insert هي التالية:

```
Insert Into TableName(Field1,Field2,..,Fieldn )
Values ('Field1Val','Field1Val',...,'FieldnVal')
```

سيقوم المتغير s من الشفرة السابقة بدور علامة التنصيص ولهذا هيئناه بالقيمة المذكورة. كما أن الحقول عبارة عن نصوص وبناء عليه فإننا نحتاج إلى علامة تنصيص قبل وبعد كل قيمة ممنوحة للحقل و هو أمر لا نحتاجه لو كانت الحقول قيما عددية.

الجديد في الأمر أننا عرفنا متغير من الصنف OleDbCommand وأعطيناه الاسم SaveInto . سنستخدم هذا الكائن لتنفيذ أمر الإضافة إلى قاعدة البيانات, كما يمكن أن نستخدمه لتنفيذ أوامر الحذف والتعديل و هو شيء سنقوم به في مراحل لاحقة .

تعمل كائنات الصنف OleDbCommand في الوضع المتصل (لماذا؟) لأنها تستطيع تغيير محتويات قاعدة البيانات كما انها تجلب معطيات من قاعدة البيانات ولهذا فان الخاصية Connection من الكائن SaveInto أخذت القيمة Conn أي تم ربطها مع كائن الاتصال . إن منح القيمة CommandType.Text للخاصية CommandType يعني أن الكائن SaveInto سينفذ عبارة SQL موجودة داخل الخاصية CommandText. نقوم بعد ذلك بأخذ القيم الموجودة داخل المكونات textBox1و textBox2 و textBox3 وتضمينها ضمن جملة SQL واضافة مايلزم من كلمات محجوزة وعلامات تنصيص والخ.

في المجموعة الثانية من الشفرة قمنا بعمل اتصال مع قاعدة البيانات ونفذنا أمر الإضافة ثم أغلقنا الاتصال .

يقوم السطر التالي بإظهار النموذج الرئيس وقد وضحنا ذلك عندما تناولنا الزرCancelBtn

```
Application.OpenForms[0].Show();
```

انظر الى السطر التالي من الشفرة :

```
(Application.OpenForms[0] as Form1).RefreshDataSet();
```

متى نستخدم الكلمة المحجوزة as ؟ عندما نطلب من كائن من صنف الأب أن يتصرف كأحد الأبناء بمعنى أننا نريد استخدام خاصية او منهج موجود في الابن و غير موجود في الأب وستتم العملية بنجاح إذا تمت تهيئة هذا المتغير بقيمة من نوع الابن(OpenForms[0] في حالتنا هذه وهو من النوع Form) أو أسندت إليه قيمة من نوع الابن . معروف أن المنهج Show هو منهج يحمل التوصيف public وموجود في كل النماذج اما المنهج RefreshDataSet هو منهج قمنا نحن بكتابة شفرته وغير موجود الا في الابناء من الصنف Form1 لذا كان لزاما علينا تحويل OpenForms[0] إلى كائن من الصنف Form1 (واكيد أن العملية ستتم بنجاح ذلك أن أول نموذج أنشيء من قبل البرنامج هو نموذج من النوعForm1) أضف الى ذلك يحمل التوصيف public أيضا لكي نتمكن من استدعاءه من داخل شفرة النموذج Form2. نعود الآن إلى النموذج Form1 ونضيف شفرةالمنهج RefreshDataSet :

```
public void RefreshDataSet()
        {
                Conn.Open();
                DataSet1.Clear();
                DataAdapter1.Fill(DataSet1,"Table1");
                Conn.Close();
        }
```

قمنا بمسح محتويات شبكة المعطيات (DataSet1)ثم ملأناها بمعطيات حديثة مجلوبة من قاعدة البيانات (من جلبها وكيف عرف البيانات المطلوب جلبها)؟ ونقصد بحديثة أن التغيرات التي اجريت من اضافة وحذف وتعديل قد تم اخذها بعين الاعتبار وستعرض للمستخدم.

فقط لاحظ إننا وضعنا شفرة الإضافة داخل كتلة من النوع try وذلك لضمان تلافي الأخطاء إذا قام احدهم بإدخال معلومات غير مقبولة (مثلا نص يحتوي على علامة تنصيص ') او غير ذلك .

قم بتشغيل البرنامج ثم انقر على الزر ADD واستمتع بإضافة سجلات إلى قاعدة البيانات الخاصة بك.

لحذف سجلات قم بالنقر مرتين على الزر Delete من النموذج Form1 واكتب الشفرة التالية:

```
private void Delete_Click(object sender, EventArgs e)
        {
            if( MessageBox.Show("Are You Sure That You Want to Delete?",
                "Warning", MessageBoxButtons.YesNo)
                ==DialogResult.Yes)

                OleDbCommand DeleteCommand = new OleDbCommand();
                DeleteCommand.Connection = Conn;
                DeleteCommand.CommandType = CommandType.Text;
                DeleteCommand.CommandText = "Delete from Table1 where
TableID=" + label5.Text;
       try
            {
                Conn.Open();
                DeleteCommand.ExecuteNonQuery();
                Conn.Close();
                MessageBox.Show("Successful Operation");
                ChangePosition();
                RefreshDataSet();
            }
            catch(Exception exp)
            {
            }

        }
```

قمنا بسؤال المستخدم لتأكيد رغبته في حذف السجل الحالي لأنه قد ينقر عن طريق الخطأ على الزر Delete فإذا أكد رغبته في حذف السجل شرعنا في خطوات الحذف . انظر إلى الشفرة السابقة ستلاحظ اننا انشأنا كائنا من النوع OleDbCommand و أعطيناه الاسم DeleteCommand ربطناه بكائن الاتصال وحددنا له الطريقة التي يتلقى بها جملة الـSQL التي سينفذها انظر إلى السطر الرابع ستجد أننا أضفنا label5.Text إلى جملة الـSQL التي سيقوم الكائن DeleteCommand بتنفيذها . لو يتذكر القارىء الكريم في بداية الدرس , أننا قمنا بإخفاء المكون label5 وذلك بجعل الخاصتين BackColor و ForceColor تأخذان القيمة Control . في الحقيقة لم نخف هذا المكون وإنما جعلنا لون النص ولون خلفية المكون لونا واحدا ولا تنسى أننا ربطنا المكون label5 مع الحقل TableID من الجدول Table1 داخل شفرة الحدث Load من النموذج Form1.

فـجملـة الـSQL التي سينفذها الكائن DeleteCommand تقول احذف السجل الذي تكون فيه قيمة الحقل (TableID) مساوية للقيمة الموجودة داخل label5.Text (لم نضف أي علامة تنصيص لماذا؟).

لماذا وضعنا الشفرة داخل كتلة من النوع Try ؟ ولماذا قمنا باستدعاء المناهج ChangePosition ؟ RefreshDataSet؟

وضعنا الشفرة داخل كتلة من النوع try لضمان تلافي الأخطاء التي قد تحدث عند إجراء عملية الحذف أما ChangePosition فمن خصائص عملية الحذف أن السجل الحالي بعد الحذف يصبح هو السجل الاول أي أن الـPosition ستاخذ القيمة 0 وهو امر يجب أن يؤخذ بعين الاعتبار عند اظهار معلومات عن السجل الحالي وقل نفس الشيء عن المنهج الـRefreshDataSet الذي سيجدد المعلومات الموجودة داخل شبكة المعطيات لان هناك سجلات حذفت ويجب إلا تظهر للمستخدم كسجلات حالية. (من المسؤول عن جعل المستخدم يرى السجلات الموجودة داخل شبكة المعطيات وليس السجلات التي في القاعدة نفسها؟ ولماذا قد تختلف المعطيات الموجودة في القاعدة عن المعطيات الموجودة في شبكة المعطيات؟).

والجواب على السؤال هو أن عملية الربط بين المكونات وشبكة المعطيات والموجودة في الحدث Load من النموذج Form1 هي المسؤولة عن جعل المستخدم معطيات الشبكة وليس معطيات القاعدة .اما الجواب على السؤال الثاني وهو أن الشبكة تعمل في الوضع

المنفصل (disconnected mode) وبالتالي فإن التغييرات التي تقع في قاعدة البيانات لن تؤثر على المعطيات الموجودة في الشبكة إلا إذا فعلنا ذلك عن عمد و هو ما نفعله عند استدعائنا للأمر RefreshDataSet.

قم بتشغيل البرنامج وتمتع بحذف السجلات واضافتها.

تعديل سجل موجود:

أضف نموذج جديد إلى البرنامج كما هو موضح بالشكل (Figure 3) و الشكل (Figure 4) ستلاحظ أن النافذة Solution Explorer أصبحت تحتوي على 3 نماذج كما هو موضح في الشكل (Figure 8). قم بتشكيل النموذج Form3 كي يصبح مثل النموذج الموضح في الشكل (Figure 9) ثم اذهب إلى الخاصية Modifiers لكل من textBox1 و textBox2 و textBox3 و label4 واجعلها تاخذ القيمة Public .

اذهب إلى الشفرة الخاصة بالنموذج Form3 بنفس الطريقة الموضحة في الشكل (Figure 6) و أضف الأسطر التالية :

Figure8

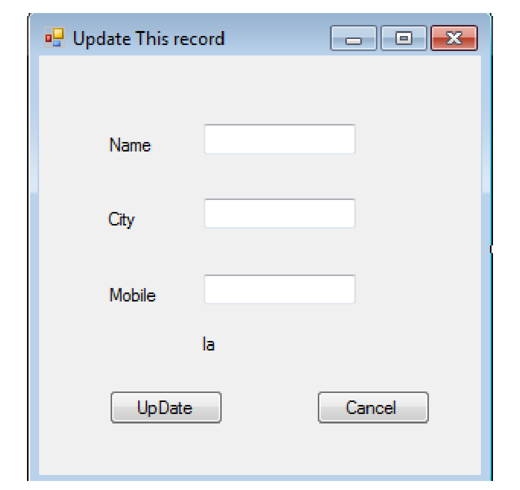


Figure9

```
public partial class Form3 : Form
    {
        //********************************
        static string ConStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
           "Data Source=MyData.mdb ";
        OleDbConnection Conn = new OleDbConnection(ConStr);
        //********************************
```

اذهب إلى النموذج Form1 وانقر مرتين على الزر Upadate وقم بكتابة الشفرة التالية:

```
private void UpdateBtn_Click(object sender, EventArgs e)
        {
            Form3 frm3 = new Form3();
            frm3.label4.Text = this.label5.Text;
            frm3.textBox1.Text = this.textBox1.Text;
            frm3.textBox2.Text = this.textBox2.Text;
            frm3.textBox3.Text = this.textBox3.Text;
            frm3.Show();
        }
```

ستضمن الشفرة السابقة نقل البيانات الموجودة في النموذج Form1 إلى النموذج Form3(ما الفائدة من جعل المكونات المذكورة سابقا تاخذ القيمة Public من اجل الخاصية Modifiers) و الآن اذهب إلى النموذج Form3 وانقر مرتين على الزر Update واكتب الشفرة التالية:

```
private void UpadateBtn_Click(object sender, EventArgs e)
        {

            string s = "'";
            OleDbCommand UpdateCommand = new OleDbCommand();
            UpdateCommand.Connection = Conn;
            UpdateCommand.CommandType = CommandType.Text;
            UpdateCommand.CommandText = "UpDate Table1 Set EName=" + s +
textBox1.Text + s +", City=" + s + textBox2.Text + s + ", Mobile=" +
+s+textBox3.Text+s+ "  where TableID="+label4.Text;
            //*******************
            try
            {
                Conn.Open();
                UpdateCommand.ExecuteNonQuery();
                Conn.Close();
                MessageBox.Show("Successful Upadate Operation");
                Application.OpenForms[0].Show();
                (Application.OpenForms[0] as Form1).RefreshDataSet();
                this.Close();
            }
            catch (Exception ex)
            {
            }

        }
```

فقط اذكر القاريء الكريم أن الصيغة العامة للعبارة Upadate في لغة الـSQL هي

Upadate TableName **Set** Fileld1='Field1Value' , Fileld2='Field2Value', …, Fileldn='FieldnValue'

Where Fileldr='FieldrValue'

طبعا نضيف علامات التنصيص إذا كانت الحقل نصيا ونحذفها إذا كان الحقل عدديا. لا يوجد هناك ما هو جديد في الشفرة السابقة حيث أننا اتبعنا نفس الأسلوب الذي طبقناه عند الإضافة حيث قمنا بتعريف كائن من النوع OleDbCommand ثم قمنا بتحديد كائن الاتصال له والطريقة التي سيتلقى بها أمر الـSQL الذي سينفذه وذكرنا له الأمر صراحة وخزناه في الخاصية CommandText وهو شيء فعلناه مرارا في شفرات سابقة. بعد أن قمنا بتنفيذ أمر الـ SQL (في أي سطر فعلنا ذلك؟) قمنا بإظهار النموذج Form1 (كيف تعرف أن Form1 هو من سيظهر؟) وجددنا البيانات الموجودة في شبكة المعطيات الموجودة في النموذج Form1 بالمنهج RefreshDataSet .

قم بتشغيل البرنامج وجرب .

البحث :

حتى هذه اللحظة قمنا بأغلب الوظائف التي تعملها برامج قواعد البيانات ولم يبق لنا البحث:

سنضيف نموذجا رابعا للبرنامج Form4 ونقوم بتنسيقه بحيث يصبح بالشكل (Figure 10) حيث يقوم المستخدم بتحديد الحقل الذي سنبحث عن السجلات من اجله, وكذلك سنقوم بتحديد سياسة البحث (تطابق تام , تطابق جزئي) وسندخل الكلمة التي نبحث عنها في المكون textBox1

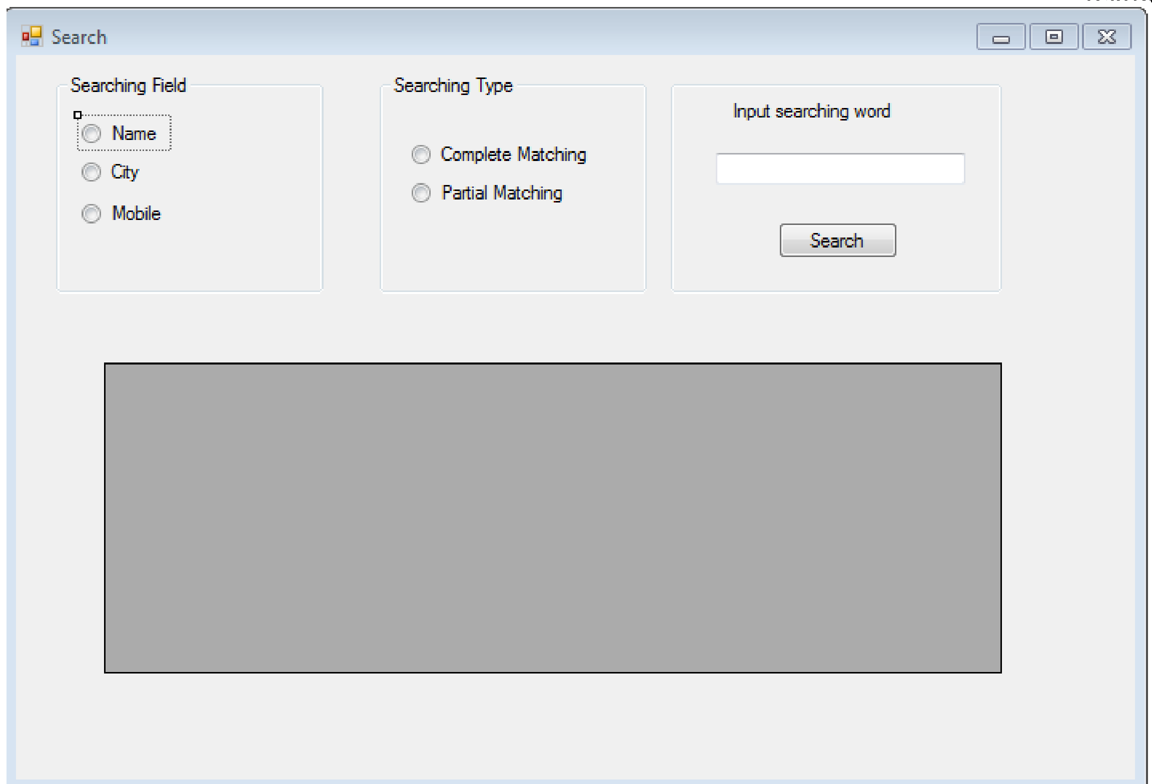


Figure10

أضفنا مكونا من الصنف DataGridView و هو من سيعرض نتائج البحث.

أضف المقدمة المعهودة إلى بداية الشفرة الخاصة بالنموذج Form4 على الصورة:

```
public partial class Form4 : Form
    {
        //============================================================
        static string ConStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
          "Data Source=MyData.mdb ";
        OleDbConnection Conn = new OleDbConnection(ConStr);
        DataSet DataSet1 = new DataSet();
        //============================================================
```

....

لاحظ معي أخي القاريء أن النموذج Form4 يحتوي على شبكة معطيات لأننا سنستعرض بيانات عملية البحث داخل هذا النموذج. سنضيف شفرتي المنهجين ExatSearch و GeneralSearch حيث المنهج يقوم الأول بالبحث عن السجلات التي تحتوي على تطابق تام بين الكلمة المدخلة والحقل الذي اخترناه بينما يقوم المنهج الثاني بالبحث عن تطابق جزئي بين الكلمة المدخلة والحقل

المحدد للبحث نقوم بتحديد الحقل عن طريق ال Radio Buttons الموجودة في المجموعة Searching Field (انظر الشكل Figure 10) ونقوم بتحديد سياسة البحث عن ال Radio Buttons الموجودة في المجموعة Searching Type. واليك أخي الكريم الشفرة.

بعد أن قمت بتنسيق النموذج كما موضح في الشكل (Figure 10) اجعل الخاصية Name من ال Radio Buttons الموجودة في المجموعة Searching Field تأخذ القيم NameRadio و CityRadio و MobileRadio محافظا على التوافق بين الخاصية Name والخاصية Text في كل مكون.ثم أضف الشفرة التالية إلى النموذج:

```
//*****************************************
private void GeneralSearch()
  {
    string SQLstr = "Select * from Table1 ";
      if (NameRadio.Checked == true)
       SQLstr = SQLstr + " where Ename Like '%" + textBox1.Text + "%'";
        else
         if (CityRadio.Checked == true)
          SQLstr = SQLstr + " where City Like '%" + textBox1.Text + "%'";
          else
           if (MobileRadio.Checked == true)
           SQLstr = SQLstr + " where Mobile Like '%" + textBox1.Text + "%'";
      try
       {
          Conn.Open();
          OleDbDataAdapter DataAdapter1 = new OleDbDataAdapter(SQLstr, Conn);
          DataAdapter1 = new OleDbDataAdapter(SQLstr, Conn);
          DataSet1.Clear();
          DataAdapter1.Fill(DataSet1, "Table1");
          Conn.Close();
          dataGridView1.DataSource = DataSet1;
          dataGridView1.DataMember = "Table1";
        }
      catch (Exception ex)
      {
      }

   }
   //*****************************************
```

تقوم الأسطر الاولى بتحديد الحقل موضوع البحث وهذا واضح في المقارنات if وتضيف اسم الحقل المحدد ضمن أمر الـSQL الذي سينفذ لجلب المعطيات من القاعدة . كذلك لاحظ أننا استخدمنا الكلمة المحجوزة Like و هي المسؤولة عن تحديد كيفية التطابق الجزئي

(ما معنى كيفية التطابق الجزئي؟ ما الدور الذي تضطلع به الكلمة Like في الـSQL؟).

اترك الأسطر المحصورة في الكتلة try للقاريء الكريم كتمرين ماعدا السطرين الاخيرين , حيث حددنا مصدر المعطيات للمكون dataGridView1 ثم حددنا الجدول الذي سيعرضه المكون الاخير ذلك أن شبكة المعطيات قد تحتوي على اكثر من جدول وعليه فلابد من تحديد الجدول الذي سيعرضه المكون.

وهذه هي الشفرة التي تخص المنهج ExactSearch بالمسؤول عن ضمان تطابق تام بين الحقل وكلمة البحث ولاتختلف بنيته الاساسية عن بنية سابقة إلا في مسالة التطابق لاحظ الطريقة التي شكلنا بها جملة الـSQL.

```
private void ExactSearch()
     {
        string SQLstr = "Select * from Table1 ";
         if (NameRadio.Checked == true)
            SQLstr = SQLstr + " where Ename = '" + textBox1.Text + "'";
```

```
            else
             if (CityRadio.Checked == true)
                SQLstr = SQLstr + " where City = '" + textBox1.Text + "'";
                else
              if (MobileRadio.Checked == true)
                 SQLstr = SQLstr + " where Mobile ='" + textBox1.Text + "'";


    try
      {
         Conn.Open();
         OleDbDataAdapter DataAdapter1 = new OleDbDataAdapter(SQLstr, Conn);
         DataAdapter1 = new OleDbDataAdapter(SQLstr, Conn);
         DataSet1.Clear();
         DataAdapter1.Fill(DataSet1, "Table1");
         Conn.Close();
         dataGridView1.DataSource = DataSet1;
         dataGridView1.DataMember = "Table1";
       }
        catch(Exception ex)
        {
        }

    }// End of ExactSearch
```

بعد ذلك اجعل Name من ال Radio Buttons الموجودة في المجموعة Searching Type تأخذ القيم PartialRadio و CompleteRadio محافظا على التوافق بين الخاصية Name والخاصية Text في كل مكون.

قم بالنقر مرتين على الزر Search من النموذج Form4 واكتب الشفرة التالية:

```
private void SearchBtn_Click(object sender, EventArgs e)
        {
            if(PartialRadio.Checked)
            {
                GeneralSearch();
            }
            else
                if (CompleteRadio.Checked)
                {

                    ExactSearch();
                }
        }
```